
Minimax-optimal semi-supervised regression on unknown manifolds

Amit Moscovich, Ariel Jaffe, Boaz Nadler

{amit.moscovich, ariel.jaffe, boaz.nadler}@weizmann.ac.il

Department of Computer Science and Applied Mathematics

Weizmann Institute of Science, Rehovot, Israel.

Abstract

We consider the problem of semi-supervised regression when the predictor variables are drawn from an unknown manifold. A simple approach to this problem is to first use both the labeled and unlabeled data to estimate the manifold geodesic distance between pairs of points, and then apply a k nearest neighbor regressor based on these distance estimates. We prove that given sufficiently many unlabeled points, this simple method which we dub *geodesic k NN regression*, achieves the optimal *finite-sample* minimax bound on the mean squared error, as if the manifold were completely specified. Furthermore, we show how this approach can be efficiently implemented requiring only $O(kN \log N)$ operations to estimate the regression function at all N labeled and unlabeled points. We illustrate this approach on two datasets with a manifold structure: indoor localization using WiFi fingerprints and facial pose estimation. For both problems we obtain better results than the popular procedure based on Laplacian eigenvectors.

1 Introduction

In recent years, many different methods for semi-supervised regression and classification have been proposed (see the surveys by [Chapelle et al. \(2006\)](#); [Zhu et al. \(2009\)](#); [Subramanya and Talukdar \(2014\)](#)). These methods have demonstrated empirical success on some data sets, whereas on others the unlabeled data does not appear to help at all. This raised two key questions of continued interest:

(i) Under what settings can the potentially huge amount of unlabeled data help with the learning process? (ii) Can we design statistically sound and computationally efficient methods that benefit from the unlabeled data?

In the theoretical study of semi-supervised learning, the following two models are commonly considered: the *cluster assumption* and the *manifold assumption*. Under the cluster assumption, instances with the same label tend to concentrate in well-defined clusters separated by low density regions ([Chapelle and Zien, 2005](#); [Rigollet, 2007](#); [Singh et al., 2009](#)). Under the manifold assumption the data points are contained in one or several low-dimensional manifolds, with nearby instances on the manifold having similar response values.

A key result involving semi-supervised learning under the manifold setting appeared in the paper by [Bickel and Li \(2007\)](#) and was further analyzed by [Lafferty and Wasserman \(2007\)](#). They proved that, as the number of labeled points tends to infinity, standard multivariate polynomial regression, using only the labeled data and without knowing the manifold structure, achieves the minimax rate for Sobolev functions. However, this asymptotic result leaves much to be desired. Intuitively, as the number of labeled points tends to infinity, the geometry of the data manifold and the sampling density on it can be accurately estimated from the labeled data alone. There is then little benefit to the availability of additional unlabeled data. Clearly, to understand the potential gains of incorporating unlabeled data when the amount of labeled data is limited, a different approach is needed. We thus turn to a finite-sample analysis. One example of such an analysis appeared in [Niyogi \(2013\)](#) who showed that for a specially constructed manifold, supervised learning is provably more difficult than semi-supervised learning. A related work by [Goldberg et al. \(2009\)](#) discussed the manifold and multi-manifold cases. In particular, they conjectured that semi-supervised learning of a Hölder function on an unknown manifold with intrinsic dimension d can achieve the finite-sample minimax bound for nonparametric regression in \mathbb{R}^d . See Section 2.1 of their paper.

In this paper we prove that when the regressed function is Lipschitz, a simple method of semi-supervised regression based on geodesic nearest neighbor averaging

ing obtains the finite-sample minimax bound when the amount of unlabeled points is sufficiently large. This settles the conjecture of Goldberg et al. (2009) for the Lipschitz case.

The regression method we consider is comprised of two parts: first, we estimate the manifold geodesic distances by shortest-path distances in a graph constructed from both the labeled and unlabeled points. The response at an unlabeled point is then estimated by averaging its k geodesic nearest labeled neighbors. We call this method *geodesic kNN regression*. Section 2 presents a description of the graph construction and nonparametric statistical methods involving geodesic nearest neighbors. Our main result, detailed in Section 3, is a proof that for a Lipschitz function on a manifold, if enough unlabeled samples are available, then with high probability this method achieves the finite-sample minimax bound. In Section 4 we discuss the computational aspects of this approach, which is very fast compared to commonly used semi-supervised methods. Furthermore, in section 4.2 we propose a simple extension to the case of inductive learning that is both fast to compute and minimax optimal. Finally in Section 5 we apply our method to a problem of indoor localization using WiFi fingerprints and an additional problem of facial pose estimation. Both of these data sets are structured as low-dimensional manifolds, so manifold-adaptive methods should perform well on them. Indeed, our results show that geodesic kNN exhibits a marked improvement compared to classical kNN, which does not utilize the unlabeled data, and also compared to the popular semi-supervised regression method of Belkin and Niyogi (2004).

2 Semi-supervised learning with geodesic distances

We consider the following framework for transductive semi-supervised learning. Given a sample of n labeled instances $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and m unlabeled instances $\mathcal{U} = \{\mathbf{x}_j\}_{j=1}^m$ from an instance space \mathcal{X} equipped with a distance function $d(\mathbf{x}, \mathbf{x}')$:

1. Construct an undirected (sparse) graph G whose vertices are the set of all labeled and unlabeled points. Pairs of close points \mathbf{x}, \mathbf{x}' are then connected by an edge with weight $w(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}, \mathbf{x}')$.
2. Compute the shortest-path graph distance $d_G(\mathbf{x}_i, \mathbf{x}_j)$ for all $\mathbf{x}_i \in \mathcal{L}$ and $\mathbf{x}_j \in \mathcal{L} \cup \mathcal{U}$.
3. Apply standard metric-based supervised learning methods, such as kNN or Nadaraya-Watson, using the computed distances d_G .

This framework generalizes the work of Bijral et al. (2011), which assumed that the samples are vectors in \mathbb{R}^D and the distance function is a power of a p -norm $\|\mathbf{x}_i - \mathbf{x}_j\|_p^q$. The use of geodesic nearest neighbors for classification was also considered by Belkin and Niyogi (2004). Specific edge selection rules include the distance-cutoff rule, whereby two points are connected by an edge if and only if their distance is below a threshold, and the symmetric k -NN rule, where every point is connected by an edge to its k nearest neighbors and vice versa. (Alamgir and von Luxburg, 2012; Ting et al., 2010)

The elegance of this framework is that it *decouples* the unsupervised and supervised parts of the learning process. It represents the geometry of the samples by a single metric d_G , thus enabling the application of any supervised learning algorithm based on a metric. For classification, a natural choice is the k nearest neighbors algorithm. For regression, one may similarly employ a k nearest neighbor regressor. Let $\text{kNN}(\mathbf{x}) \subseteq \mathcal{L}$ denote the set of k (or less) nearest *labeled* neighbors to \mathbf{x} , where the distance is measured by the metric d_G . We call this method *geodesic kNN regression*. Its estimator, with uniform weights, is given by

$$\hat{f}(\mathbf{x}) = \frac{1}{|\text{kNN}(\mathbf{x})|} \sum_{(\mathbf{x}_i, y_i) \in \text{kNN}(\mathbf{x})} y_i. \quad (1)$$

In the next section we analyze the statistical properties of this regressor under the manifold assumption. In particular, we prove that if enough unlabeled points are available, \hat{f} obtains the minimax bound on the mean squared error.

3 Statistical analysis under the manifold assumption

Before analyzing the geodesic kNN regressor of Eq. (1) we first recall some classical results from the theory of nonparametric estimation. Let $f : \mathbb{R}^D \rightarrow \mathbb{R}$ be some function. Given n noisy samples of f , there are many ways of constructing regression estimators \hat{f} . For every choice of regressor \hat{f} we may analyze its mean squared error at a point $\mathbf{x} \in \mathbb{R}^D$, $\text{MSE}(\hat{f}, \mathbf{x}) := \mathbb{E}[(f(\mathbf{x}) - \hat{f}(\mathbf{x}))^2]$, where the expectation is taken over draws of n samples. It can be shown that for *any* estimator \hat{f} and any point $\mathbf{x} \in \mathbb{R}^D$, there is some Lipschitz function f such that $\text{MSE}(\hat{f}, \mathbf{x}) \geq cn^{-\frac{2}{2+D}}$ for some constant $c > 0$ that depends only on the Lipschitz constant and the noise level. The term $n^{-\frac{2}{2+D}}$ is thus termed a *finite-sample minimax lower bound* on the MSE at a point. Many results of this type are known, involving various measures of risk and classes of functions. See Tsybakov (2009); Györfi et al. (2002).

Using standard nonparametric regression methods such as Nadaraya-Watson or kNN regression, one is guaranteed an upper bound on the MSE that is also of the form $c'n^{-\frac{2}{2+d}}$. Hence these methods are termed *minimax optimal* for estimating a Lipschitz function.

In Theorem 1 below we prove that if the data set is sampled from a Lipschitz function on a manifold, then given a sufficient number of *unlabeled* points, the MSE of the geodesic kNN regressor is upper-bounded by $c''n^{-\frac{2}{2+d}}$ where c'' is some constant and d is the intrinsic dimension of the manifold rather than the extrinsic dimension D . Hence it is minimax-optimal and adaptive to the geometry of the unknown manifold.

Before stating our main result, we first introduce some notation and describe our assumptions. For a general background on smooth manifolds, see for example the book by Lee (2012). We assume that the data manifold $\mathcal{M} \subseteq \mathbb{R}^D$ is a compact smooth manifold of intrinsic dimension d , possibly with boundaries and corners. We further assume that \mathcal{M} is geodesically convex, i.e. that every two points in \mathcal{M} are connected by a geodesic curve. These conditions will be needed later, in order to apply the theorems of Tenenbaum et al. (2000). We denote by $d_{\mathcal{M}}(\mathbf{x}, \mathbf{x}')$ the length of the shortest path between two points in \mathcal{M} , the diameter of \mathcal{M} by $\text{diam}(\mathcal{M}) := \sup_{\mathbf{x}, \mathbf{x}'} d_{\mathcal{M}}(\mathbf{x}, \mathbf{x}')$ and the manifold-ball of points around \mathbf{x} by $B_{\mathbf{x}}(r) := \{\mathbf{x}' \in \mathcal{M} : d_{\mathcal{M}}(\mathbf{x}, \mathbf{x}') < r\}$.

We consider a standard nonparametric regression model, $Y = f(X) + \mathcal{N}(0, \sigma^2)$ where $X \in \mathbb{R}^D$ is drawn according to a measure μ on \mathcal{M} and $f : \mathcal{M} \rightarrow \mathbb{R}$ is an L -Lipschitz function on \mathcal{M} , i.e.

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{M} : |f(\mathbf{x}) - f(\mathbf{x}')| \leq L d_{\mathcal{M}}(\mathbf{x}, \mathbf{x}'). \quad (2)$$

In order to bound the MSE at a point \mathbf{x} we need the following condition on the measure μ to hold,

Definition 1. A measure μ is (R, Q) -lower-bounded at \mathbf{x} if for every radius $r \leq R$, $\mu(B_{\mathbf{x}}(r)) \geq Qr^d$.

This condition simply means that the measure of a small ball around \mathbf{x} has the typical volume expansion property for dimension d .

3.1 Main result

We consider the geodesic kNN regressor \hat{f} , computed for a set of points on a manifold $\mathcal{M} \subseteq \mathbb{R}^D$ with a known intrinsic dimension d , using either the distance-cutoff or the symmetric-kNN edge selection rule and with edge weights $w(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$. Our main theorem bounds the expected MSE of $\hat{f}(\mathbf{x})$ at a fixed point $\mathbf{x} \in \mathcal{M}$. The expectation is to be understood in the following manner: the graph G is constructed from the

point \mathbf{x} and a random draw of n labeled points and m unlabeled points. Then $\hat{f}(\mathbf{x})$ is computed using the shortest-path graph distances d_G .

Theorem 1 (Minimax optimality). *Let $\mathbf{x} \in \mathcal{M}$ be a point such that μ is (R, Q) -lower-bounded at \mathbf{x} . There exist functions $C_{\mathcal{M}, \mu}(\epsilon, \delta)$ and $c(R, Q, \delta)$ such that for any choice of $\epsilon, \delta \in (0, 1)$ if the number of unlabeled points satisfies $m \geq C_{\mathcal{M}, \mu}(\epsilon, \delta)$ then*

$$\mathbb{E} \left[(\hat{f}(\mathbf{x}) - f(\mathbf{x}))^2 | A \right] \leq (2L^2 c(R, Q, \delta) + \sigma^2) n^{-\frac{2}{2+d}}$$

where \hat{f} is the geodesic kNN regressor with $k = \lceil n^{\frac{2}{2+d}} \rceil$ and A is an event that holds with probability $\geq 1 - \epsilon$.

Remark 1. $C_{\mathcal{M}, \mu}(\epsilon, \delta)$ is the minimal number of points needed to obtain, with probability $\geq 1 - \epsilon$, a sufficiently dense point cover of \mathcal{M} such that the graph distances between pairs of points approximate their manifold distances up to a factor of $1 \pm \delta$. This number is typically exponential in the intrinsic dimension.

Remark 2. Kpotufe (2011, Theorem 1) proved that with data sampled from an unknown manifold, even classical (supervised) kNN based on Euclidean distances achieves the finite-sample minimax bound up to log factors. However, his result requires $O(\log n)$ labeled points in a small Euclidean ball around \mathbf{x} . This is different from our result that merely requires a dense sample of *unlabeled* points, but holds for any n .

Let $X_1, \dots, X_n \in \mathcal{M}$ be a sample of n labeled points and let $X_{n+1}, \dots, X_{n+m} \in \mathcal{M}$ be a sample of m unlabeled points. For a fixed point $\mathbf{x} \in \mathcal{M}$, let G be a graph constructed from $\{\mathbf{x}\} \cup \{X_1, \dots, X_{n+m}\}$. We denote the closest labeled point to \mathbf{x} according to either the graph or the manifold distance by

$$X_G^{(1,n)}(\mathbf{x}) := \underset{X_i \in \{X_1, \dots, X_n\}}{\operatorname{argmin}} d_G(\mathbf{x}, X_i)$$

$$X_{\mathcal{M}}^{(1,n)}(\mathbf{x}) := \underset{X_i \in \{X_1, \dots, X_n\}}{\operatorname{argmin}} d_{\mathcal{M}}(\mathbf{x}, X_i).$$

The following lemma bounds the expected squared manifold distance to the closest labeled graph vertex.

Lemma 1. *Let $\mathbf{x} \in \mathcal{M}$ be some point for which μ is (R, Q) -lower-bounded. There exist functions $C_{\mathcal{M}, \mu}(\epsilon, \delta)$ and $c(R, Q, \delta)$ such that for any choice of $\epsilon, \delta \in (0, 1)$, if the number of unlabeled points satisfies $m \geq C_{\mathcal{M}, \mu}(\epsilon, \delta)$ then*

$$\mathbb{E} \left[d_{\mathcal{M}}^2(\mathbf{x}, X_G^{(1,n)}(\mathbf{x})) | A \right] \leq c(R, Q, \delta) n^{-\frac{2}{2+d}} \quad (3)$$

where A is an event that holds with probability $\geq 1 - \epsilon$.

Proof. By main theorems B and C in the supplementary of Tenenbaum et al. (2000) there is a function $C_{\mathcal{M}, \mu}(\epsilon, \delta)$ such that for any choice of $\epsilon, \delta \in (0, 1)$

if we are given $m \geq C_{\mathcal{M},\mu}(\epsilon, \delta)$ random samples $X_1, \dots, X_m \sim \mu$ and we construct a graph according to either the distance-cutoff rule or the symmetric-kNN rule then with probability at least $1 - \epsilon$ the following inequalities hold for all X_i, X_j in the sample,

$$1 - \delta \leq d_G(X_i, X_j)/d_{\mathcal{M}}(X_i, X_j) \leq 1 + \delta. \quad (4)$$

We denote by A the event that all of these inequalities are satisfied. From now on we assume that A holds. By applying Eq. (4) twice, we obtain an upper bound on the manifold distance to the closest labeled point in the graph.

$$\begin{aligned} d_{\mathcal{M}}(\mathbf{x}, X_{\mathcal{M}}^{(1,n)}(\mathbf{x})) &\geq \frac{d_G(\mathbf{x}, X_{\mathcal{M}}^{(1,n)}(\mathbf{x}))}{1 + \delta} \\ &\geq \frac{d_G(\mathbf{x}, X_G^{(1,n)}(\mathbf{x}))}{1 + \delta} \geq \frac{1 - \delta}{1 + \delta} d_{\mathcal{M}}(\mathbf{x}, X_G^{(1,n)}(\mathbf{x})) \end{aligned} \quad (5)$$

Now we apply the well-known equality for a non-negative random variable $\mathbb{E}[X] = \int_0^\infty \Pr[X > r] dr$,

$$\begin{aligned} \mathbb{E}[d_{\mathcal{M}}^2(\mathbf{x}, X_G^{(1,n)}(\mathbf{x}))] &= \int \Pr[d_{\mathcal{M}}^2(\mathbf{x}, X_G^{(1,n)}(\mathbf{x})) > r] dr \\ &= \int_0^{\text{diam}(\mathcal{M})^2} \Pr[d_{\mathcal{M}}(\mathbf{x}, X_G^{(1,n)}(\mathbf{x})) > \sqrt{r}] dr \\ &\leq \int_0^{\text{diam}(\mathcal{M})^2} \Pr[d_{\mathcal{M}}(\mathbf{x}, X_{\mathcal{M}}^{(1,n)}(\mathbf{x})) > \frac{1 - \delta}{1 + \delta} \sqrt{r}] dr \end{aligned} \quad (6)$$

where the last inequality follows from Eq. (5). Note that \mathcal{M} is compact, therefore $\text{diam}(\mathcal{M}) < \infty$. By definition, $\Pr[d_{\mathcal{M}}(\mathbf{x}, X_{\mathcal{M}}^{(1,n)}(\mathbf{x})) > t] = (1 - \mu(B_{\mathbf{x}}(t)))^n$. Recall that μ is (R, Q) -lower-bounded at \mathbf{x} , hence for any $t \leq R$ we have $\mu(B_{\mathbf{x}}(t)) \geq Qr^d$ and therefore

$$\Pr[d_{\mathcal{M}}(\mathbf{x}, X_{\mathcal{M}}^{(1,n)}(\mathbf{x})) > t] \leq (1 - Qr^d)^n \quad (7)$$

We denote $\lambda := (1 + \delta)/(1 - \delta)$ and split the domain of the integral in Eq. (6) in two: $I_1 = [0, \lambda^2 R^2]$ and $I_2 = [\lambda^2 R^2, \text{diam}(\mathcal{M})^2]$. For every $r \in I_1$ we have $\sqrt{r}(1 - \delta)/(1 + \delta) < R$ so we may apply inequality (7).

$$\begin{aligned} &\int_0^{\lambda^2 R^2} \Pr[d_{\mathcal{M}}(\mathbf{x}, X_{\mathcal{M}}^{(1,n)}(\mathbf{x})) > \sqrt{r}/\lambda] dr \\ &\leq \int_0^{\lambda^2 R^2} (1 - Q\lambda^{-d}r^{d/2})^n dr \leq \int_0^{\lambda^2 R^2} e^{-Q\lambda^{-d}r^{d/2}n} dr. \end{aligned}$$

We split this domain of integration into segments of length $n^{-2/d}$. Since the integrand is monotone-decreasing in r we can bound each integral by the value of the integrand at the left endpoint.

$$\begin{aligned} &\int_0^{\lambda^2 R^2} e^{-Q\lambda^{-d}r^{d/2}n} dr \\ &= \int_0^{n^{-2/d}} e^{-Q\lambda^{-d}r^{d/2}n} dr + \int_{n^{-2/d}}^{2n^{-2/d}} e^{-Q\lambda^{-d}r^{d/2}n} dr + \dots \\ &\leq n^{-\frac{2}{d}} \left(1 + e^{-Q\lambda^{-d}} + e^{-Q\lambda^{-d}\sqrt{2^d}} + e^{-Q\lambda^{-d}\sqrt{3^d}} + \dots\right) \end{aligned}$$

The parenthesized sum converges to some constant which depends on Q, d and δ (this can be proved using the Raabe-Duhamel test). We bound the integral over I_2 using the fact that the integrated probability is decreasing and apply Eq. (7).

$$\begin{aligned} &\int_{\lambda^2 R^2}^{\text{diam}(\mathcal{M})^2} \Pr[d_{\mathcal{M}}(\mathbf{x}, X_{\mathcal{M}}^{(1,n)}(\mathbf{x})) > \sqrt{r}/\lambda] dr \\ &\leq \text{diam}(\mathcal{M})^2 \cdot \Pr[d_{\mathcal{M}}(\mathbf{x}, X_{\mathcal{M}}^{(1,n)}(\mathbf{x})) > R] \\ &\leq \text{diam}(\mathcal{M})^2 \cdot e^{-QR^{d/2}n} \end{aligned}$$

As a function of n this is negligible with respect to the integral over I_1 , hence Eq. (3) follows. \square

We are now ready to prove Theorem 1. The following proof is an adaptation of Theorem 6.2 of Györfi et al. (2002) to the setting of a Lipschitz function on a manifold. It differs from their proof in three aspects:

(i) The data points are sampled from a manifold \mathcal{M} , rather than Euclidean space. (ii) The distances to the labeled points are only known approximately. (iii) We derive a bound on the MSE at a given point \mathbf{x} rather than the (weaker) bound on the expected MSE for a random point $\mathbf{x} \sim \mu$.

Proof. As in the proof of Lemma 1, we denote by A the event that the inequalities of Eq. (4) hold for all i, j . In the rest of this proof we assume A holds.

Denote $X_{1\dots n} = \{X_1, \dots, X_n\}$. By the bias-variance decomposition and the law of total variance,

$$\begin{aligned} \mathbb{E}[(\hat{f}(\mathbf{x}) - f(\mathbf{x}))^2] &= \text{bias}^2(\mathbf{x}) + \text{Var}(\hat{f}(\mathbf{x})) \\ &= \text{bias}^2(\mathbf{x}) + \mathbb{E}[\text{Var}(\hat{f}(\mathbf{x})|X_{1\dots n})] + \text{Var}(\mathbb{E}[\hat{f}(\mathbf{x})|X_{1\dots n}]). \end{aligned} \quad (8)$$

We will bound these three terms separately. Denote by $X_G^{(i,n)}(\mathbf{x})$ the i -th closest labeled point to \mathbf{x} (in terms of the graph distance), by $Y_G^{(i,n)}(\mathbf{x})$ its response and by $\eta_G^{(i,n)}(\mathbf{x}) := Y_G^{(i,n)}(\mathbf{x}) - f(X_G^{(i,n)}(\mathbf{x}))$ the noise. For any random variable Z , we have $\mathbb{E}^2[Z] \leq \mathbb{E}[Z^2]$. Applying this, we obtain a bound on the squared bias.

$$\begin{aligned} \text{bias}^2(\mathbf{x}) &= \mathbb{E}^2\left[\frac{1}{k} \sum_{i=1}^k f(X_G^{(i,n)}(\mathbf{x})) - f(\mathbf{x})\right] \\ &\leq \mathbb{E}\left[\left(\frac{1}{k} \sum_{i=1}^k (f(X_G^{(i,n)}(\mathbf{x})) - f(\mathbf{x}))\right)^2\right] \\ &\leq \mathbb{E}\left[\left(\frac{1}{k} \sum_{i=1}^k L \cdot d_{\mathcal{M}}(X_G^{(i,n)}(\mathbf{x}), \mathbf{x})\right)^2\right]. \end{aligned} \quad (9)$$

Randomly split the data set X_1, \dots, X_n into disjoint subsets S_1, \dots, S_{k+1} , such that $|S_1| = \dots = |S_k| = \lfloor \frac{n}{k} \rfloor$

and S_{k+1} contains the remaining elements. Denote the closest element to \mathbf{x} from the S_i by $S_i(\mathbf{x})$. Clearly,

$$\sum_{i=1}^k d_{\mathcal{M}} \left(X_G^{(i,n)}(\mathbf{x}), \mathbf{x} \right) \leq \sum_{i=1}^k d_{\mathcal{M}} (S_i(\mathbf{x}), \mathbf{x}). \quad (11)$$

We plug this back into Eq. (10) and apply Jensen's inequality.

$$\begin{aligned} \text{bias}^2(\mathbf{x}) &\leq L^2 \mathbb{E} \left[\left(\frac{1}{k} \sum_{i=1}^k d_{\mathcal{M}} (S_i(\mathbf{x}), \mathbf{x}) \right)^2 \right] \\ &\leq L^2 \mathbb{E} \left[\frac{1}{k} \sum_{i=1}^k d_{\mathcal{M}}^2 (S_i(\mathbf{x}), \mathbf{x}) \right] = L^2 \mathbb{E} [d_{\mathcal{M}}^2 (S_1(\mathbf{x}), \mathbf{x})]. \end{aligned} \quad (12)$$

The set S_1 is simply a random draw of $\lfloor \frac{n}{k} \rfloor$ points, so by Lemma 1,

$$\text{bias}^2(\mathbf{x}) \leq L^2 c(R, Q, \delta) \lfloor \frac{n}{k} \rfloor^{-2/d}. \quad (13)$$

We now bound the second term in Eq. (8). By the definition of the geodesic kNN regressor,

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^k \frac{Y_G^{(i,n)}(\mathbf{x})}{k} = \sum_{i=1}^k \frac{f(X_G^{(i,n)}(\mathbf{x})) + \eta_G^{(i,n)}(\mathbf{x})}{k}.$$

Conditioned on X_1, \dots, X_n the terms $f(X_G^{(i,n)}(\mathbf{x}))$ are constant. The noise η has zero mean and is independent of the draw of X_1, \dots, X_n . Therefore

$$\text{Var} \left(\hat{f}(\mathbf{x}) | X_{1..n} \right) = \frac{\sigma^2}{k}. \quad (14)$$

For the third term in (8) we note that for any real random variable Z and any $c \in \mathbb{R}$, we have $\text{Var}(Y) = \mathbb{E}[(Y - \mathbb{E}[Y])^2] \leq \mathbb{E}[(Y - c)^2]$. Hence,

$$\begin{aligned} \text{Var} \left(\mathbb{E} [\hat{f}(\mathbf{x}) | X_{1..n}] \right) &= \text{Var} \left(\frac{1}{k} \sum_{i=1}^k f(X_G^{(i,n)}(\mathbf{x})) \right) \\ &\leq \mathbb{E} \left[\left(\frac{1}{k} \sum_{i=1}^k f(X_G^{(i,n)}(\mathbf{x})) - f(\mathbf{x}) \right)^2 \right]. \end{aligned}$$

This is equal to Eq. (9). So we obtain the same result as the bound on the squared bias.

$$\text{Var} \left(\mathbb{E} [\hat{f}(\mathbf{x}) | X_{1..n}] \right) \leq L^2 c(R, Q, \delta) \lfloor \frac{n}{k} \rfloor^{-2/d} \quad (15)$$

To conclude, by substituting equations (13), (14) and (15) into Eq. (8), we obtain

$$\mathbb{E} \left[(\hat{f}(\mathbf{x}) - f(\mathbf{x}))^2 \right] \leq 2L^2 c(R, Q, \delta) \lfloor \frac{n}{k} \rfloor^{-2/d} + \frac{\sigma^2}{k}.$$

The theorem follows by setting $k = \lceil n^{\frac{2}{2+d}} \rceil$. \square

4 Computation of geodesic kNN

Theorem 1 shows that the geodesic kNN regressor outlined in Section 2 is minimax optimal. In this section we show how it can also be computed efficiently, assuming that the graph has already been constructed (more on this in Section 4.1 below). Computing $\hat{f}(\mathbf{x})$ for all points in a dataset reduces to the following algorithmic problem: Let $G = (V, E)$ be a weighted undirected graph and let $\mathcal{L} \subseteq V$ be a subset of labeled vertices. How can we efficiently find the k nearest labeled neighbors of every vertex in the graph? Denote $n = |\mathcal{L}|$, $N = |V|$. A simple approach to this problem is to first apply Dijkstra's algorithm from each of the labeled points, forming an $n \times N$ matrix of all pairwise shortest graph distances $d_G(s, v)$, where $s \in \mathcal{L}$ and $v \in V$. The k nearest labeled vertices of the j^{th} vertex correspond to the k smallest cells in the j^{th} column. The runtime of this method is $O(nN \log N + n|E|)$ (Dasgupta et al., 2006).

For $k = 1$, where one computes the single nearest labeled vertex to every vertex in a graph, the result is known as the graph Voronoi diagram, with the labeled vertices acting as the centers of the Voronoi cells. A fast algorithm for this problem was developed by Erwig (2000). Algorithm 1 that we present here is a generalization of his approach for any $k \geq 1$. Before describing it, we briefly recall Dijkstra's shortest path algorithm: Given a seed vertex $s \in V$, Dijkstra's algorithm keeps, for every vertex $v \in V$, an upper bound on $d_G(s, v)$, denoted $u[v]$, initialized to 0 if $v = s$ and to $+\infty$ otherwise. At every iteration, the vertex v_0 with the lowest upper bound is *visited*: For every neighbor v of v_0 , if $u[v_0] + w(v_0, v) < u[v]$, then the current upper bound $u[v]$ is lowered. v_0 is never visited again.

The basic idea behind Algorithm 1 can be described as running n instances of Dijkstra's algorithm "simultaneously" from all labeled vertices. This is combined with an early stopping rule whenever k paths from different labeled vertices have been found.

As in Dijkstra's algorithm, Algorithm 1 uses a priority queue based on a Fibonacci heap with the 3 standard operations: insert, pop-minimum and decrease-key. We use decrease-or-insert as a shorthand for decreasing the key of an element if it is stored in the queue, and otherwise inserting it. Instead of storing vertices in the priority queue, as in Dijkstra's algorithm, Algorithm 1 stores pairs (seed, v) keyed by dist , where dist is the current upper bound on $d_G(\text{seed}, v)$. In the supplementary we prove that whenever $(\text{dist}, \text{seed}, v)$ is popped from the queue, we have $\text{dist} = d_G(\text{seed}, v)$. At every iteration, the pair (seed, v_0) with the lowest upper bound is *visited*: we examine every neighbor v of v_0 and possibly update the current upper bound of

$d_G(\text{seed}, v)$ using a decrease-or-insert operation.

Algorithm 1 Geodesic k nearest labeled neighbors

Input: An undirected weighted graph $G = (V, E, w)$ and a set of labeled vertices $\mathcal{L} \subseteq V$.

Output: For every $v \in V$ a list $kNN[v]$ with the k nearest labeled vertices to v and their distances.

```

 $Q \leftarrow \text{PriorityQueue}()$ 
for  $v \in V$  do
     $kNN[v] \leftarrow \text{Empty-List}()$ 
     $S_v \leftarrow \phi$ 
    if  $v \in \mathcal{L}$  then
         $\text{insert}(Q, \text{element} = (v, v), \text{priority} = 0)$ 
    end if
end for
while  $Q \neq \phi$  do
     $(\text{seed}, v_0, \text{dist}) \leftarrow \text{pop-minimum}(Q)$ 
     $S_{v_0} \leftarrow S_{v_0} \cup \{\text{seed}\}$ 
    if  $\text{length}(kNN[v_0]) < k$  then
        append  $(\text{dist}, \text{seed})$  to  $kNN[v_0]$ 
        for all  $v \in \text{neighbors}(v_0)$  do
            if  $\text{seed} \notin S_v$  then
                 $\text{decrease-or-insert}(Q, \text{element} = (\text{seed}, v),$ 
                     $\text{priority} = \text{dist} + w(v_0, v))$ 
            end if
        end for
    end if
end while
    
```

For every vertex $v_0 \in V$, pairs of the form (seed, v_0) are visited at most k times. In the supplementary we prove the correctness of Algorithm 1 and show that due to this early stopping rule, it is significantly more efficient than the naïve approach.

Independently of our work, this algorithm was recently described by Har-Peled (2016) for the $\mathcal{L} = V$ case. Furthermore, a clever optimization of the priority queues was proposed which yields a runtime that can be bounded by

$$O(k|V| \log |V| + k|E|). \quad (16)$$

In the supplementary, we present this as Algorithm 2 and compare its empirical runtime with Algorithm 1.

As for memory requirements, both Algorithm 1 and Algorithm 2 use memory bounded by the minimum of $O(n|V|)$ and $O(k|E|)$. The first bound follows from the fact that Q cannot have more than $|\mathcal{L}| \times |V|$ elements. The second follows since every vertex v is visited at most k times and may insert up to $\deg(v)$ neighbors into Q .

Remark 3. Bijral et al. (2011) also proposed a variant of Dijkstra’s algorithm. However, their method is

an improvement of *single-source* Dijkstra in the setting of a dense graph constructed from points in R^D , whereas the methods we discuss here compute paths from *multiple sources* and are applicable to any graph.

4.1 Notes on the graph construction time

For the construction of G , the straightforward approach is to compute the distances between all pairs of points in time $O(D|V|^2)$. In light of Eq. (16) this may take much longer than actually computing geodesic kNN on the graph G . One way to reduce the running time is to store all of the data points in a k -d tree (Bentley, 1975) a ball tree (Omohundro, 1989) or some other data structure for spatial queries and then find nearby neighbors for every point. These data structures are suitable for constructing both distance-cutoff graphs and symmetric-kNN graphs from low-dimensional data. For high-dimensional data, several works have appeared in recent years which propose fast methods of constructing approximate kNN graphs (Zhang et al., 2013; Wang et al., 2013). The running time of these methods is $O(D|V| \log |V|)$ multiplied by some constant which is empirically small. Combining these constructions with the fast algorithm for computing geodesic kNN regression yields a total running time of $O((k+D)|V| \log |V|)$. This is much faster than many other semi-supervised methods, which involve expensive computations such as eigendecomposition (Belkin and Niyogi, 2004) or matrix inversion (Zhu et al., 2003).

4.2 Inductive learning

In this section we consider the inductive setting of semi-supervised learning, where one wishes to evaluate $\hat{f}(\mathbf{x})$ for a new instance \mathbf{x} . The naïve approach is to add the new point to our data set and recompute geodesic kNN on all points. However, this is clearly wasteful. Instead we propose a much simpler method: assume we have already computed regression estimates $\hat{f}(\mathbf{x}_i)$ for a set of points $\mathcal{L} \cup \mathcal{U}$. Given a new data point $\mathbf{x} \notin \mathcal{L} \cup \mathcal{U}$, we first find its nearest neighbor \mathbf{x}^* from either the labeled or unlabeled set. This can be typically done in sublinear time either using specialized data structures for spatial queries (Omohundro, 1989; Bentley, 1975) or by employing approximate nearest neighbor methods (Andoni and Indyk, 2006). Then our regression estimate at \mathbf{x} is simply $\hat{f}(\mathbf{x}^*)$. As the following theorem shows, this simple method achieves the minimax bound.

Theorem 2. Assume that the conditions of Theorem 1 are satisfied and that μ is (R, Q) -lower-bounded for all points in \mathcal{M} . Let $\mathbf{x} \in \mathcal{M}$ be a point and denote its

Euclidean nearest neighbor by

$$\mathbf{x}^* := \operatorname{argmin}_{\mathbf{x}' \in \mathcal{L} \cup \mathcal{U}} \|\mathbf{x} - \mathbf{x}'\|.$$

Then

$$\mathbb{E} \left[(\hat{f}(\mathbf{x}^*) - f(\mathbf{x}))^2 | A \right] \leq (3L^2 c(R, Q, \delta) + \sigma^2) n^{-\frac{2}{2+d}} \quad (17)$$

where A is an event that holds with probability $\geq 1 - \epsilon$.

Proof. As before, we define the event A to be true iff all the inequalities of Eq. (4) are satisfied. For any $a, b \in \mathbb{R}$, $(a + b)^2 \leq (a + b)^2 + (a - b)^2 = 2a^2 + 2b^2$. This gives a bound on the mean squared error.

$$\begin{aligned} & \mathbb{E} \left[(\hat{f}(\mathbf{x}^*) - f(\mathbf{x}))^2 | A \right] \\ &= \mathbb{E} \left[\left((\hat{f}(\mathbf{x}^*) - f(\mathbf{x}^*)) + (f(\mathbf{x}^*) - f(\mathbf{x})) \right)^2 | A \right] \\ &\leq 2\mathbb{E} \left[(\hat{f}(\mathbf{x}^*) - f(\mathbf{x}^*))^2 | A \right] + 2\mathbb{E} \left[(f(\mathbf{x}^*) - f(\mathbf{x}))^2 | A \right]. \end{aligned}$$

We bound these terms separately. By Theorem 1,

$$\mathbb{E} \left[(\hat{f}(\mathbf{x}^*) - f(\mathbf{x}^*))^2 | A \right] \leq (2L^2 c(R, Q, \delta) + \sigma^2) n^{-\frac{2}{2+d}}.$$

Note that $\mathbf{x}^* = X_G^{(1, n+m)}(\mathbf{x})$, thus by the fact that f is L -Lipschitz and Lemma 1, we have

$$\begin{aligned} & \mathbb{E} \left[(f(\mathbf{x}^*) - f(\mathbf{x}))^2 | A \right] \leq \mathbb{E} \left[L^2 d_{\mathcal{M}}^2(X_G^{(1, n+m)}(\mathbf{x}), \mathbf{x}) | A \right] \\ &\leq L^2 c(R, Q, \delta) (n + m)^{-\frac{2}{d}} \leq L^2 c(R, Q, \delta) n^{-\frac{2}{2+d}}. \end{aligned}$$

□

5 Applications

5.1 Indoor localization using Wi-Fi signals

One motivation for our work is the problem of estimating the location of a mobile device in a closed environment using its Wi-Fi signature as received by a wireless router. This problem is gaining considerable interest in recent years due to its many potential applications, such as indoor navigation inside large commercial spaces (Liu et al., 2007). In indoor settings, the signal received by the router is a superposition of multiple reflections of the same source, which differ in their arrival time, direction and intensity. This limits the use of classic outdoor positioning methods such as triangulation, which require a direct line-of-sight between the transmitting device and the receiver.

A common approach for tackling this problem, known as *fingerprinting* in the signal processing community, is based on nearest-neighbor search. First, a labeled set

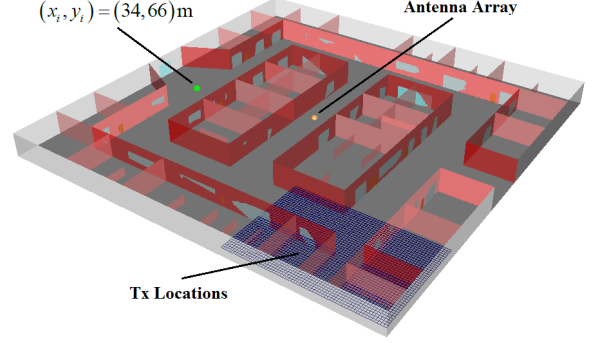


Figure 1: 3D model of a $80 \times 80m \times 5m$ floor.

$\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is collected, where $y_i \in \mathbb{R}^2$ is the location of the transmitter and \mathbf{x}_i is a feature vector that depends in some way on the received signal. The location of new instances is then estimated via non-parametric regression methods such as k nearest neighbors.

For applications requiring high accuracy, recording and maintaining a suitable labeled data set may be prohibitively expensive. On the other hand, collecting vast amounts of unlabeled data may be done simply by recording the Wi-Fi signals of various devices moving through the venue. Indoor localization is thus a natural application for semi-supervised methods. Moreover, the space of feature vectors is parameterized by a 2 or 3 dimensional position. Thus, we expect manifold-based methods to perform well in this task. To test this empirically, we used two data sets of indoor localization: a simulated and a real data set. A brief description follows. See the supplementary for details.

Simulated data: This data consists of 802.11 Wi-Fi signals in an artificial $80m \times 80m$ indoor office environment generated by Kupershtein et al. (2013) using a 3D radio wave propagation software, see Figure 1.

Real data: These are actual 802.11 signals, recorded by a Wi-Fi router placed roughly in the middle of a $27m \times 33m$ office, see Figure 6 of the supplementary.

The Signal Subspace Projection (SSP) of Kupershtein et al. (2013) and Jaffe and Wax (2014) is used as the fingerprint for localization. It is based on the assumption that signals received from close locations maintain similar properties of differential delays and directions of arrival. In our experiments, the SSP features are 48×48 projection matrices, where the projected subspace is 10-dimensional. We use the Frobenius norm as the distance metric and construct a symmetric-4NN graph as described in Section 2. For more details on the datasets and SSP features, see the supplementary.

We compare our semi-supervised geodesic kNN regressor to Laplacian eigenvector regression (Belkin and

Table 1: Mean accuracy of kNN vs. geodesic kNN vs. Laplacian eigenbasis regression on the real data set

Labeled grid	#labeled	kNN	GNN	Lap
1.5m	73	1.49m	1.11m	1.36m
2.0m	48	2.27m	1.49m	1.65m
3m	23	3.41m	2.41m	2.79m

(Niyogi, 2004). As a baseline we also applied classic kNN regression, using only the labeled samples and optimizing over k . Figure 2 shows the median localization error on the simulated data set as a function of the number of unlabeled locations, where the labeled points are placed on a fixed grid. Specifically, for the geodesic kNN regressor we used $k = 7$ and exponentially decaying weights such that the weight of the i -th neighbor is proportional to $1/2^i$. For Laplacian eigenvector regression, we optimized over the number of eigenvectors by taking the best outcome after repeating the experiment with 10%, 20%, 30%, 40% and 50% of the labeled points.

Table 1 shows the mean localization error on the real data set for different densities of labeled points. The results on both the simulated and real datasets show a clear advantage for the geodesic kNN regressor. As expected, the improvement shown by the semi-supervised methods increases with the number of unlabeled locations. Moreover geodesic kNN regression is much faster to compute than the Laplacian eigenvector regressor, see Table 2 in the supplementary.

5.2 Facial pose estimation

We illustrate the performance of geodesic kNN on another regression problem, using the **faces** data set where the predicted value is the left-right angle of a face image.¹ This data set contains 698 greyscale images of a single face rendered at different angles and lighting. The instance space is the set of all 64×64 images whereas the intrinsic manifold dimension is 3. For our benchmark, we computed the ℓ_1 distance between all pairs of images and constructed a symmetric 4-NN graph. For the geodesic kNN algorithm, the edge weights were set to the ℓ_1 distances and k was set to 1. For Laplacian eigenvector regression we used binary weights and set the number of eigenvectors to 20% of the number of labeled points. This is a common rule-of-thumb, and gave good results over the whole range. Figure 3 shows that geodesic kNN performs uniformly better than the nearest neighbor regressor and also outperforms the semi-supervised Laplacian regressor.

¹<http://isomap.stanford.edu/datasets.html>

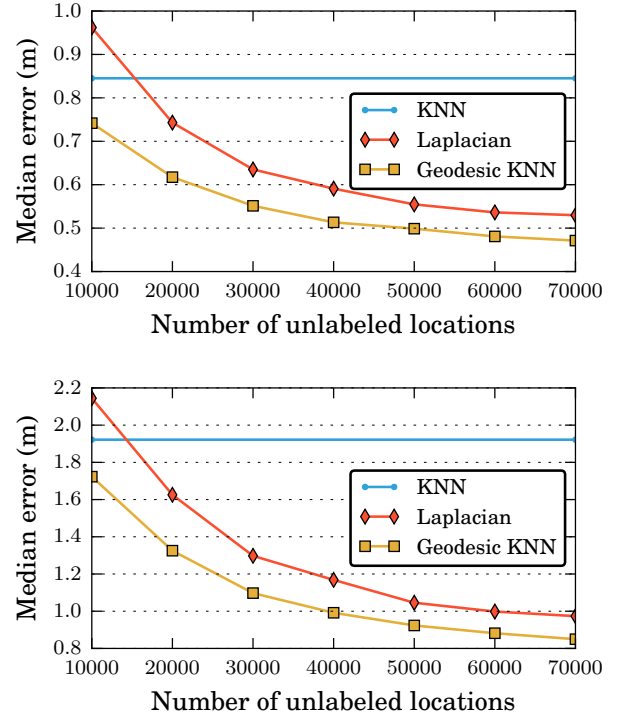


Figure 2: Median localization error vs. number of unlabeled points. Upper panel: 1600 labeled points placed on a regular grid with a side length of 2m. Lower panel: 400 labeled points on a 4m grid.

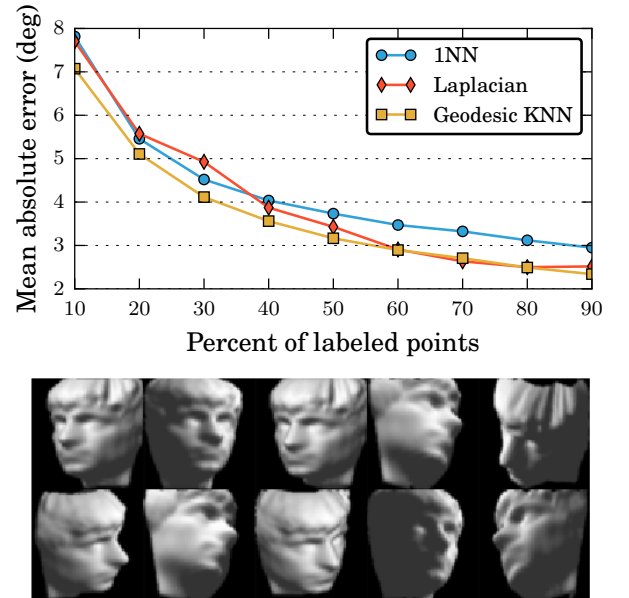


Figure 3: (top) Mean prediction error for the left-right angle of the face. (bottom) Sample images from the **faces** data set, showing different poses and lighting.

Supplementary Material

6 Fast transductive geodesic nearest neighbors

In this section we provide more details regarding the efficient computation of geodesic kNN for all vertices in a graph. We begin by proving the correctness of Algorithm 1 and analyzing its runtime. Then, in Section 6.3 we present Algorithm 2 by Har-Peled (2016) which has a tighter bound on the asymptotic running time. Finally, in Section 6.4 we test the empirical running time of these algorithms on the simulated WiFi data set.

6.1 Proof of correctness

We start with some auxiliary definitions and lemmas regarding shortest paths. Given an undirected and weighted graph G , we denote paths in G by $v_i \rightarrow v_j \rightarrow \dots \rightarrow v_k$ or simply $v_i \rightsquigarrow v_k$ when the context makes it clear what path we are referring to. The *length* of a path is the sum of its edge weights,

$$w(v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_m) = \sum_{i=1}^{m-1} w(v_i, v_{i+1}).$$

The length of the shortest (geodesic) path between two vertices $v, v' \in V$ is denoted by $d_G(v, v')$. A path $v \rightsquigarrow v'$ is called a *shortest path* if $w(v \rightsquigarrow v') = d_G(v, v')$. The following lemma is the key to the correctness of Algorithms 1 and 2.

Lemma 2. *Let $v \in V$ be a vertex and let s be its j -th nearest labeled vertex. If $s \rightsquigarrow u \rightsquigarrow v$ is a shortest path then $s \in \text{NLV}(u, j)$, where $\text{NLV}(u, j)$ is the set of j nearest labeled vertices to u .*

Proof. Assume by contradiction that there are j labeled nodes s_1, \dots, s_j such that

$$\forall i : d_G(s_i, u) < d_G(s, u).$$

Then

$$\begin{aligned} d_G(s_i, v) &\leq d_G(s_i, u) + d_G(u, v) \quad (\text{triangle inequality}) \\ &< d_G(s, u) + d_G(u, v) \quad (\text{by assumption}) \\ &= d_G(s, v). \end{aligned} \tag{18}$$

where the last equality is derived from the assumption that u is on a shortest path between s and v . Eq. (18) implies that the vertices s_1, \dots, s_j are all closer than s to v , which contradicts the assumption. \square

We now continue to the main part of the proof.

Lemma 3. *For every triplet $(dist, seed, v_0)$ popped from Q there is a path $seed \rightsquigarrow v_0$ of length $dist$.*

Proof. We prove the claim by induction on the elements inserted into Q .

Base of the induction: The first L inserts correspond to the labeled vertices $\{(0, s, s) : s \in \mathcal{L}\}$. For these triplets, once popped out of Q the claim holds trivially.

Induction step: Any triplet inserted into Q or updated is of the form $(dist + w(v, v_0), seed, v)$ where $(dist, seed, v_0)$ was previously inserted into Q . Hence, by the induction hypothesis there exists a path $seed \rightsquigarrow v_0$ of length $dist$. Since v is a neighbor of v_0 with edge weight $w(v_0, v)$, there exists a path $seed \rightsquigarrow v_0 \rightarrow v$ of length $w(seed \rightsquigarrow v_0) + w(v_0 \rightarrow v) = dist + w(v_0, v)$. \square

Lemma 4. *The distances popped from Q in the main loop form a monotone non-decreasing sequence.*

Proof. This follows directly from the fact that Q is a minimum priority queue and that the edge weights are non-negative, hence future insertions or updates will have a priority that is higher or equal to that of an existing element in the queue. \square

Lemma 5. *Every time a triplet $(dist, seed, v_0)$ is popped from Q , the following conditions hold*

1. *If $seed \in \text{NLV}(v_0, k)$ then $dist = d_G(seed, v_0)$.*
2. *All pairs $(s, v) \in \mathcal{L} \times V$ that satisfy $d_G(s, v) < dist$ and $s \in \text{NLV}(v, k)$ are in the visited set.*

Proof. We prove both claims simultaneously by induction on the popped triplets.

Base of the induction: The first L triplets are equal to $\{(0, s, s) : s \in \mathcal{L}\}$. Part 1 holds since $d_G(s, s) = 0$. Part 2 holds because there are no paths shorter than 0.

Induction step:

Part 1. (If $seed \in \text{NLV}(v_0, k)$ then $dist = d_G(seed, v_0)$)

By Lemma 3, $dist$ is the length of an actual path, so it cannot be smaller than the shortest path length $d_G(seed, v_0)$. Assume by contradiction that $d_G(seed, v_0) < dist$. There is some *shortest* path $seed \rightsquigarrow v_p \rightarrow v_0$ of length $d_G(seed, v_0)$ (v_p may be equal to $seed$, but not to v_0). Clearly

$d_G(\text{seed}, v_p) \leq d_G(\text{seed}, v_0) < \text{dist}$ and by Lemma 3.1, $\text{seed} \in \text{NLV}(v_p, k)$. Thus by Part 2 of the induction hypothesis $(\text{seed}, v_p) \in \text{visited}$. This implies that (seed, v_p) was visited in a previous iteration. Note that it follows from $\text{seed} \in \text{NLV}(v_p, k)$ and from Lemma 4 that during the visit of (seed, v_p) the condition $\text{length}(k\text{NN}[v_p]) < k$ was true. Therefore the command $\text{dec-or-insert}(Q, d_G(\text{seed}, v_p) + w(v_p, v_0), \text{seed}, v_0)$ should have been called, but because $\text{seed} \rightsquigarrow v_p \rightarrow v_0$ is a shortest path, $d_G(\text{seed}, v_p) + w(v_p, v_0) = d_G(\text{seed}, v_0)$, leading to the conclusion that the triplet $(d_G(\text{seed}, v_0), \text{seed}, v_0)$ must have been inserted into Q in a previous iteration, which leaves two options:

1. Either $(d_G(\text{seed}, v_0), \text{seed}, v_0)$ was never popped from Q , in that case it should have been popped from Q in the current iteration instead of $(\text{dist}, \text{seed}, v_0)$. Contradiction.
2. The triplet $(d_G(\text{seed}, v_0), \text{seed}, v_0)$ was popped from Q in a previous iteration. However this implies a double visit of (seed, v_0) , which is impossible due to the use of the *visited* set.

Part 2. (All pairs $(s, v) \in \mathcal{L} \times V$ that satisfy $d_G(s, v) < \text{dist}$ and $s \in \text{NLV}(v, k)$ are contained in *visited*)

Let $(s, v) \in \mathcal{L} \times V$ be a pair of vertices that satisfies $d_G(s, v) < \text{dist}$ and $s \in \text{NLV}(v, k)$. Assume by contradiction that $(s, v) \notin \text{visited}$. Let $s \rightsquigarrow v' \rightarrow v'' \rightsquigarrow v$ be a shortest path such that $(s, v') \in \text{visited}$ but $(s, v'') \notin \text{visited}$. Such v', v'' must exist since $(s, s) \in \text{visited}$ and by our assumption $(s, v) \notin \text{visited}$. We note that s may be equal to v' and v'' may be equal to v . Since $(s, v') \in \text{visited}$, some triplet (d, s, v') was popped from Q in a previous iteration and by Part 1 of the induction hypothesis $d = d_G(s, v')$. By Lemma 3.1 we know that $s \in \text{NLV}(v', k)$, therefore during the visit of (s, v') the condition $\text{length}(k\text{NN}[v']) < k$ was true. Following this, there must have been a call to $\text{decrease-key-or-insert}(Q, d_G(s, v') + w(v', v''), s, v'')$. Now,

$$\begin{aligned} d_G(s, v') + w(v', v'') & \\ = d_G(s, v'') & \quad (s \rightsquigarrow v' \rightarrow v'' \text{ is a shortest path}) \\ \leq d_G(s, v) & \quad (s \rightsquigarrow v'' \text{ is a subpath}) \end{aligned}$$

Hence the pair (s, v'') was previously stored in Q with distance $d_G(s, v'') \leq d_G(s, v) < \text{dist}$. Since $(s, v'') \notin \text{visited}$, it should have been present in Q with a smaller distance than (s, v) as key, thus $(d_G(s, v''), s, v'')$ should have been popped instead of $(\text{dist}, \text{seed}, v_0)$, contradiction. \square

Finally, we are ready to state the theorem that Algorithm 1 produces correct output, namely that its output for every vertex $v \in V$ is indeed the set of its k nearest labeled points, as measured by the geodesic graph distance.

Theorem 3. *Let $G = (V, E, w)$ be a graph with non-negative weights. For every vertex $v \in V$ let \mathcal{L}_v denote the set of labeled vertices in the connected component of v and let $\ell_v = \min\{k, |\mathcal{L}_v|\}$. Then*

1. *Algorithm 1 stops after a finite number of steps.*
2. *Once stopped, for every $v \in V$ the output list $k\text{NN}[v]$ is of the form $k\text{NN}[v] = [(d_G(s_1, v), s_1), \dots, (d_G(s_{\ell_v}, v), s_{\ell_v})]$ where s_1, \dots, s_{ℓ_v} are the nearest labeled vertices, sorted by their distance to v .*

Proof. Part 1 follows trivially from the fact that every pair (seed, v_0) can be popped at most once and part 2 follows from Lemma 5 and an induction on the popped triplets (d, s, v) . \square

6.2 Transductive runtime analysis of Algorithm 1

Lemma 2 explains why Algorithm 1 can stop exploring the neighbors of vertices whose k nearest labeled neighbors were found. As Theorem 4 shows, this can lead to dramatic runtime savings.

Theorem 4. *Given a graph $G = (V, E)$ with n labeled vertices, the runtime of Algorithm 1 is bounded by $O(k|E| + N_p \log |V|)$ where N_p is the total number of pop-minimum operations, which satisfies $N_p \leq \min\{n|V|, k|E|\}$.*

Proof. Recall that in a priority queue based on a Fibonacci heap all operations cost $O(1)$ amortized time except pop-minimum which costs $\log |Q|$. The runtime is dominated by the cost of all pop-minimum operations, plus the total cost of traversing the neighbors of the examined vertices. The latter takes $O(k|E|)$ time. Denote the total number of pop-minimum operations by N_p . We derive two different bounds on N_p . Every time a (seed, v) pair is popped from Q , it is added to the *visited* set, which prevents future insertions of that pair into Q . Hence, each pair $(\text{seed}, v) \in \mathcal{L} \times V$ may be popped at most once from Q , which implies that $N_p \leq n|V|$. In addition, N_p is bounded by the number of insertions into Q . First, there are n insertions during the initialization phase. Then, for each vertex $v_0 \in V$, the "if $\text{length}(k\text{NN}[v_0]) < k$ " clause can hold true at most k times for that vertex. Each time, the neighbors of v_0 are examined and up to $\deg(v_0)$ neighbors are inserted into Q . This yields the second bound, $N_p \leq n + k|E| = O(k|E|)$. \square

6.3 Saving unneeded extractions

Algorithm 1 keeps a priority queue with pairs $(seed, v) \in \mathcal{L} \times V$. However, once a vertex v has been visited from k different seed nodes, we no longer need to process it further, thus any later pop operations which involve v are a waste of CPU cycles. Conceptually, once v is visited for the k^{th} time, we would like to purge the queue of all pairs (s, v) , but this is expensive since every pop operation costs $\log |Q|$. Instead we use an idea by Har-Peled (2016), which we detail in Algorithm 2. For every vertex $v \in V$ we keep a separate priority queue Q_v , which will be disabled once v is visited k times. A global priority queue Q is maintained that keeps the lowest element of each of the local queues Q_v . Now popping an element involves popping some v from Q and then popping Q_v . This is followed by an insert of the new minimum element from Q_v into Q .

Algorithm 2 Geodesic k nearest labeled neighbors with faster priority queue handling

Input: An undirected weighted graph $G = (V, E, w)$ and a set of labeled vertices $\mathcal{L} \subseteq V$.

Output: For every $v \in V$ a list $kNN[v]$ with the k nearest labeled vertices to v and their distances.

```

 $Q \leftarrow \text{PriorityQueue}()$ 
for  $v \in V$  do
     $Q_v \leftarrow \text{PriorityQueue}()$ 
     $kNN[v] \leftarrow \text{Empty-List}()$ 
     $S_v \leftarrow \phi$ 
    if  $v \in \mathcal{L}$  then
         $\text{insert}(Q, \text{element} = v, \text{priority} = 0)$ 
         $\text{insert}(Q_v, \text{element} = v, \text{priority} = 0)$ 
    end if
end for
while  $Q \neq \phi$  do
     $(v_0, \text{dist}) \leftarrow \text{pop-minimum}(Q)$ 
     $(seed, \text{dist}) \leftarrow \text{pop-minimum}(Q_{v_0})$ 
     $S_{v_0} \leftarrow S_{v_0} \cup \{seed\}$ 
    append  $(seed, \text{dist})$  to  $kNN[v_0]$ 
    if  $\text{length}(kNN[v_0]) < k$  and  $Q_{v_0} \neq \phi$  then
         $(newseed, \text{newdist}) \leftarrow \text{minimum}(Q_{v_0})$ 
         $\text{insert}(Q, \text{element} = v_0, \text{priority} = \text{newdist})$ 
    end if
    for all  $v \in \text{neighbors}(v_0)$  do
        if  $\text{length}(kNN[v]) < k$  and  $seed \notin S_v$  then
             $\text{decrease-or-insert}(Q_v, \text{element} = seed, \text{priority} = \text{dist} + w(v_0, v))$ 
             $\text{decrease-or-insert}(Q, \text{element} = v, \text{priority} = \text{dist} + w(v_0, v))$ 
        end if
    end for
end while

```

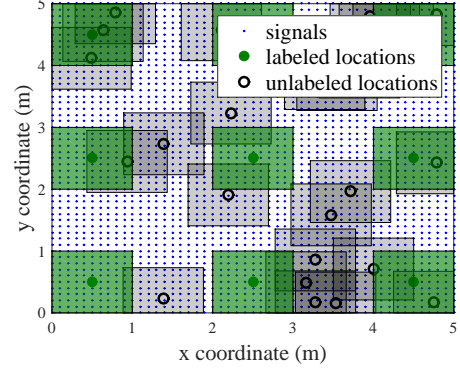


Figure 4: Schematic of the labeled and unlabeled point generation for the simulated data set. Labeled locations (green circles) were placed on a $4m$ grid, whereas the unlabeled locations (grey circles) were placed at random.

Theorem 5. Given a graph $G = (V, E)$ with n labeled vertices, the runtime of Algorithm 1 is bounded by $O(k|E| + k \log |V|)$ where N_p is the total number of pop-minimum operations, which satisfies $N_p \leq \min\{n|V|, k|E|\}$.

Proof. The runtime of Algorithm 2 is bounded by $O(k|E| + N_p \log |V|)$ where N_p is the number of pop-minimum operations. Let $v \in V$ be a vertex. Pairs of the form $(seed, v)$ may be popped at most k times. hence $N_p \leq k|V|$. \square

An immediate corollary of Theorem 4 is that for a graph $G = (V, E)$ of bounded degree d , the runtime of Algorithm 2 is $O(kd|V| \log |V|)$. In contrast, the runtime of the naïve approach based on multiple Dijkstra runs is $O(n|V| \log |V| + nd|V|)$. Comparing these two formulas, we see that major speedups are obtained in typical cases where $n \gg kd$. As we illustrate empirically in the following Section, these speedups are very large in practice, even on graphs of moderate size.

6.4 Empirical runtime comparison

We compare the runtime of Algorithms 1 and 2 to the naïve method of running Dijkstra’s algorithm from each of the labeled points. To make the comparison meaningful we implemented all of these algorithms in Python, using a similar programming style and the same heap data structure. The running times were measured for the simulated WiFi signals data set. Figure 5 shows the relative speedup (measured in seconds) of both algorithms compared to multiple Dijkstra runs. Interestingly, while Algorithm 2 has better asymptotic guarantees than Algorithm 1, both show similar speedups on this particular data set.

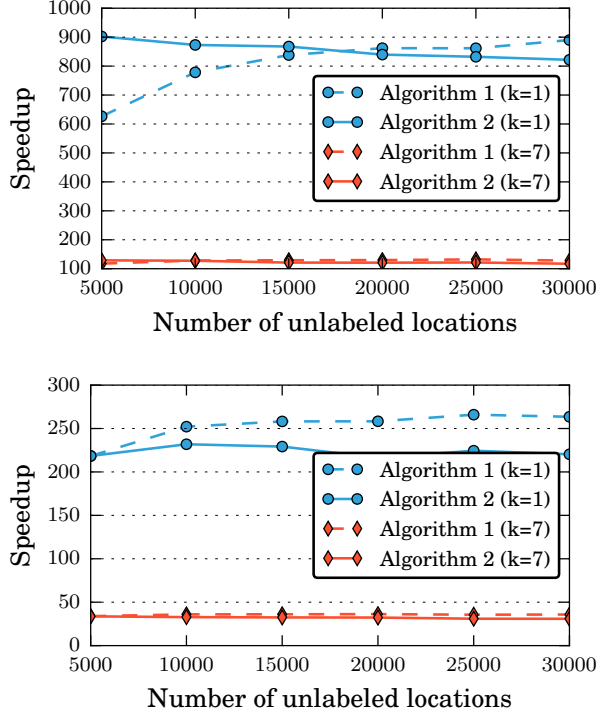


Figure 5: Relative speedup computing the geodesic 1NN and 7NN of all points in a graph using Algorithms 1 and 2. The speedup of both algorithms is compared to the naïve approach of running Dijkstra’s algorithm from each labeled point. (left panel) A total of 1600 labeled locations are placed on a 2m square grid. (right panel) A total of 400 labeled locations are on a 4m grid.

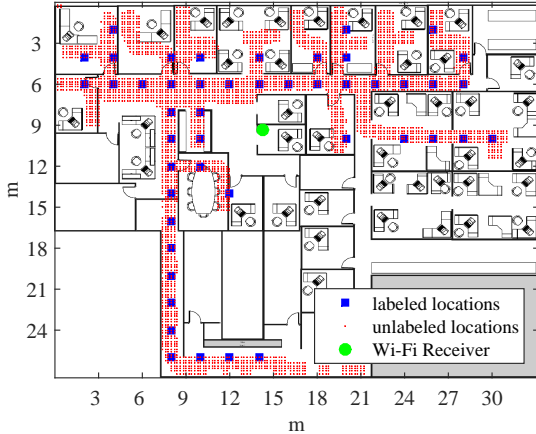


Figure 6: Schematic of the mapped areas in the real data set

Table 2: Runtime of Geodesic 7-NN vs. time to compute Laplacian eigenvectors

#unlabeled	G7NN	Laplacian	Graph build
1000	2.3s	7.6s	9s
10000	7s	195s	76s
100000	56s	114min	66min

Table 2 compares the runtime of the geodesic 7NN method to the runtime of computing the top eigenvectors of the Laplacian matrix, using the simulated indoor localization data set with labeled locations every 2m. The number of eigenvectors was chosen to be 320, which is equal to 20% of the number of labeled points. Computing the geodesic nearest neighbors by using Algorithm 1 is several orders of magnitude faster than computing the eigenvectors. This is despite the fact that the eigenvector computation is performed using the highly optimized Intel® Math Kernel Library whereas the geodesic nearest neighbor computation uses a simple Python implementation. We expect an efficient implementation of geodesic kNN to be at least 10 times faster. For completeness, the third column shows the graph construction time, using the naïve $O(n^2)$ algorithm.

7 Indoor localization details

7.1 Dataset description

Simulated data: This data consists of 802.11 Wi-Fi signals in an artificial yet realistic environment generated by Kupershtein et al. (2013) using a 3D radio wave propagation software. The environment is an $80m \times 80m$ floor. In its center is a Wi-Fi router with $p = 6$ antennas. See Figure 1. At various locations $(x, y) \in \mathbb{R}^2$, $N = 8$ consecutive samples of a Wi-Fi packet’s (constant) preamble are recorded, at equally spaced time intervals of $50\mu s$. The samples are stored in a complex-valued vector $s_{x,y} \in \mathbb{C}^{pN}$ which we refer to as the *signal* received from location (x, y) . The simulated signals were generated on a dense $0.1m$ grid covering the entire area of the floor.

Real data: This data consists of actual 802.11 signals, recorded by a Wi-Fi router with $p = 6$ antennas placed approximately in the middle of a $27m \times 33m$ office, see Figure 6. The transmitter was a tablet connected to the router via Wi-Fi. The signal vector of each location (x, y) was sampled $N = 8$ times from every antenna. The transmitter locations were entered manually by the operator. For both the labeled and unlabeled locations, we first generated a square grid covering the area of the office, and then kept only the locations that contained received signals. For the la-

beled points, we repeated the experiments with several grid sizes ranging from $1.5m$ to $3m$. The location of the WiFi router is marked by the green circle.

7.2 Feature extraction and distance metric for the SSP method

The Signal Subspace Projection method is based on the assumption that signals originating from nearby locations are high dimensional vectors contained in a low dimensional subspace. Hence, signals originating from nearby locations are contained in a low dimensional subspace. The subspace around each location is used as its signature. Specifically, the signature P_ℓ of a location $\ell = (x, y) \in \mathbb{R}^2$ is computed as follows. First, the covariance matrix of the signals in the proximity of ℓ is computed, using all the signals in the dataset that are inside a $1m$ square centered around ℓ (see Figure 4),

$$R_\ell := \sum_{\ell' \in \mathbb{R}^2: \|\ell - \ell'\|_\infty < 0.5m} s_{\ell'} s_{\ell'}^*$$

where $s_{\ell'}^*$ denotes the Hermitian transpose of the column vector $s_{\ell'} \in \mathbb{C}^{pN}$. Next, we compute the n_{pc} leading eigenvectors of R_ℓ , forming a matrix $V_\ell \in \mathbb{C}^{pN \times n_{pc}}$. The SSP signature is the projection matrix onto the space spanned by these eigenvectors, $P_\ell := V_\ell V_\ell^*$. In our experiments, we picked $n_{pc} = 10$, though other choices in the range $\{8, \dots, 12\}$ gave results that are almost as good. The distance between pairs of locations is defined as the Frobenius norm of the difference of their projection matrices, $d_{i,j} := \|P_{\ell_i} - P_{\ell_j}\|_F$. gave us results that are on par with the best methods of Jaffe and Wax (2014).

For the simulated dataset, to mimic how a real-world semi-supervised localization system might work, we generated the labeled locations on a regular square grid, whereas the unlabeled locations were randomly distributed over the entire area of the floor (see Figure 4). For the real dataset, due to physical constraints, labeled and unlabeled points were not placed on a regular grid. Then, we created a symmetric kNN graph by connecting every point x_i with its k_G closest neighbors, with corresponding weights given by $w_{i,j} = 1 + \epsilon d_{i,j}$. Here $\epsilon > 0$ is a small constant which gives preference to paths with smaller $d_{i,j}$. Experimentally using the symmetric kNN graph construction with $k = 4$ gave slightly better results than choosing $k \in \{3, 5, 6\}$ for both the geodesic kNN regressor and for the Laplacian eigenvector regressor.

References

Alamgir, M. and von Luxburg, U. (2012). Shortest path distance in random k-nearest neighbor

graphs. *International Conference on Machine Learning (ICML)*.

- Andoni, A. and Indyk, P. (2006). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *The IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–468.
- Belkin, M. and Niyogi, P. (2004). Semi-Supervised Learning on Riemannian Manifolds. *Machine Learning*, 56:209–239.
- Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.
- Bickel, P. J. and Li, B. (2007). Local polynomial regression on unknown manifolds. In *Tomography, Networks and Beyond*, pages 177–186. Institute of Mathematical Statistics.
- Bijral, A. S., Ratliff, N., and Srebro, N. (2011). Semi-supervised Learning with Density Based Distances. *Association for Uncertainty in Artificial Intelligence*, (1):43–50.
- Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. MIT Press Cambridge.
- Chapelle, O. and Zien, A. (2005). Semi-Supervised Classification by Low Density Separation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 57–64.
- Dasgupta, S., Papadimitriou, C. H., and Vazirani, U. (2006). *Algorithms*. McGraw-Hill, Inc.
- Erwig, M. (2000). The graph Voronoi diagram with applications. *Networks*, 36:156–163.
- Goldberg, A. B., Zhu, X., Singh, A., Xu, Z., and Nowak, R. D. (2009). Multi-manifold semi-supervised learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Györfi, L., Kohler, M., Krzyżak, A., and Walk, H. (2002). *A distribution-free theory of nonparametric regression*. Springer Series in Statistics. Springer-Verlag, New York.
- Har-Peled, S. (2016). Computing the k Nearest Neighbors for all Vertices via Dijkstra. *ArXiv e-prints*.
- Jaffe, A. and Wax, M. (2014). Single-Site Localization via Maximum Discrimination Multipath Fingerprinting. *IEEE Transactions on Signal Processing*, 62(7):1718–1728.
- Kpotufe, S. (2011). k -NN Regression Adapts to Local Intrinsic Dimension. In *Neural Information Processing Systems (NIPS)*.
- Kupershtein, E., Wax, M., and Cohen, I. (2013). Single-site emitter localization via multipath fingerprinting. *IEEE Transactions on Signal Processing*, 61(1):10–21.

- Lafferty, J. and Wasserman, L. (2007). Statistical analysis of semi-supervised regression. In *Neural Information Processing Systems (NIPS)*.
- Lee, J. M. (2012). *Introduction to Smooth Manifolds*. Springer New York.
- Liu, H., Darabi, H., Banerjee, P., and Liu, J. (2007). Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 37(6):1067–1080.
- Niyogi, P. (2013). Manifold Regularization and Semi-supervised Learning: Some Theoretical Analyses. *Journal of Machine Learning Research (JMLR)*, 14(1):1229–1250.
- Omohundro, S. M. (1989). Five Balltree Construction Algorithms. Technical report.
- Rigollet, P. (2007). Generalization error bounds in semi-supervised classification under the cluster assumption. *Journal of Machine Learning Research (JMLR)*, 8:1369–1392.
- Singh, A., Nowak, R., and Zhu, X. (2009). Unlabeled data: Now it helps, now it doesn’t. *Neural Information Processing Systems (NIPS)*, pages 1513–1520.
- Subramanya, A. and Talukdar, P. P. (2014). Graph-Based Semi-Supervised Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(4):1–125.
- Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science (New York, N. Y.)*, 290(5500):2319–2323.
- Ting, D., Huang, L., and Jordan, M. I. (2010). An Analysis of the Convergence of Graph Laplacians. In *International Conference on Machine Learning (ICML)*, pages 1079–1086.
- Tsybakov, A. B. (2009). *Introduction to nonparametric estimation*. Springer New York.
- Wang, D., Shi, L., and Cao, J. (2013). Fast algorithm for approximate k-nearest neighbor graph construction. In *2013 IEEE 13th International Conference on Data Mining Workshops*.
- Zhang, Y.-M., Huang, K., Geng, G.-g., and Liu, C.-L. (2013). Fast kNN graph construction with locality sensitive hashing. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 660–674.
- Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *International Conference on Machine Learning (ICML)*.
- Zhu, X., Goldberg, A. B., Brachman, R., and Dietterich, T. (2009). *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers.